N0ARY/BBS
Sysop Manual

Version 3

Architectural Overview
Daemon Interfaces

First Draft

Bob Arasmith, N0ARY
November 15, 1993

Introduction

The N0ARY BBS was created to answer a need in the packet radio community. The traditional packet bbs is very user hostile and short on functionality. They require the user to be knowledgable on the use of packet and the bbs system which most new packet users are not. The N0ARY bbs addresses this issue in the following ways:

* An artificial intelligence (AI) style user interface. Users can use commands found on other bbss as well as "talking" to the bbs. The request "list all bulletins about earthquake since monday" can be typed verbatim and will be understood by the bbs.

* All commands can be typed in the entirety or abbreviated to the minimum number of characters to be unique in the current context. The above command could be abbreviated to "la b earthquake sin mon". The order is also not important, "lb sin mon earthquake all".

* There is also redundancy in commands. To exit the bbs the user can type bye, quit, logout, logoff, goodbye or exit.

* The user also has complete control over his interface with the bbs. He can have messages listed in ascending or descending order, have his packet mail forwarded to internet/compuserve/mcimail/etc. He can also specify how many lines his terminal can display and the bbs will automatically page output. Keyboard macros can be programmed so complex commands or chains of commands can be issued with a single keystroke or automatically at login. The bbs can automatically append signatures to outgoing messages (.signature), send message received confirmation (.vacation), supports regular expressions in all search fields. Users can even specify filters to control what kind of messages will be displayed to them.

* Integrated "White Pages" lookup. The user is discouraged from supplying too much information. The bbs will try to complete all addresses if the user doesn't supply the information. It will also alert the user of differences between what he asked for and what the bbs thinks is correct.

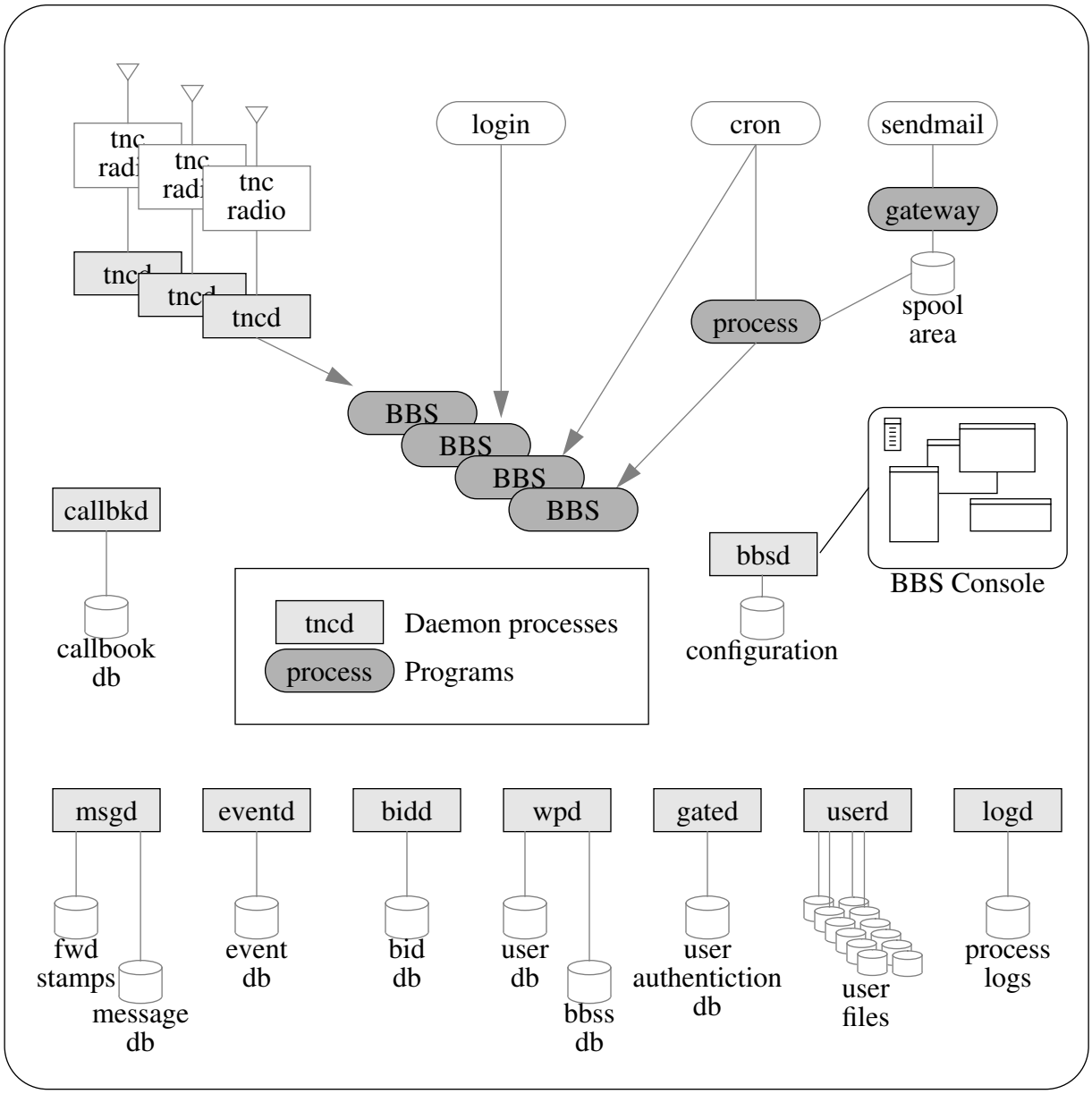
* Integrated on-line callbook. When a user connects to the bbs for the first time the bbs calls them by name. It will also fill in most of their users fields from the callbook entry. Users can lookup calls or search for amateurs by partial calls, names, geographic locations, etc.

The user interface makes the bbs very easy and fun to use. The bbs went on-line in the summer of 1990 and now has over 1200 users, the average bbs size in this area is between 30 and 100. The disadvantage to this bbs is that it runs on workstations running unix. Not that common in the amateur community.

This document describes the internals of the bbs and it's associated processes. The bbss user interface is not discussed fully in this document. It is referenced only where it is need to make things more understandable. This document is for sysop consumption only. We will start with a basic overview of the system and move into particulars of the different pieces.

Architecture

The bbs program itself is simply a command parser. It determines what the user wants, forms requests, issues these requests to the service responsible for them and displays the results to the user.

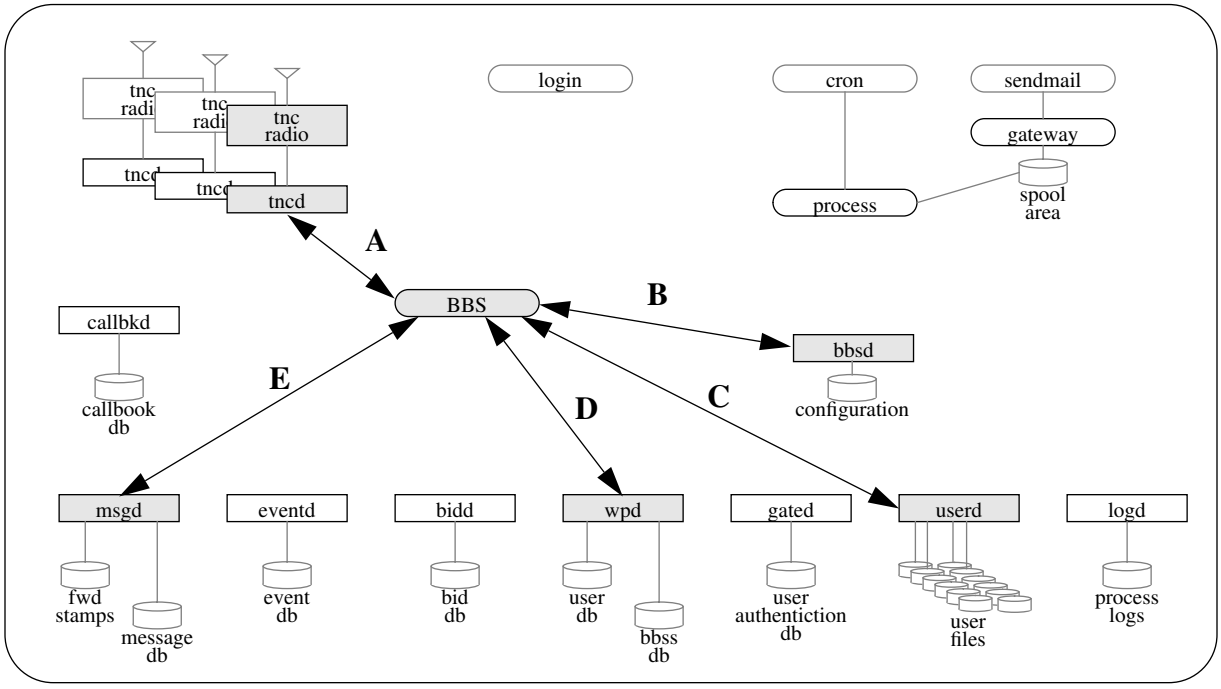


A bbs process can be spawned by either one of the tnc daemons, the gateway mail processing program, a cron, or via the system login process. Any number of bbs processes can be executing concurrently. The tnc daemons are theoretically capable of handling any number of simultaneous users but the real life limit is about 6 on a 1200 baud channel. Any number of tncd processes can be resident, one per physical tnc/radio combination.

The bbsd daemon controls the execution of all other bbs related programs. It starts by reading a configuration file that defines most of the characteristics of the environment and then starts all the bbs daemons. When any of the bbs family of programs start they first register with the bbsd process to get permission to run. They can then make requests of bbsd to learn about the system configuration; where their data files are held, what times to use for aging of data, the callsign of the bbs, etc.

The sysop can check on the status of the bbs system at anytime via the bbs motif GUI or by telneting to the bbsd daemon directly. He can then dynamically change configuration variables, lock ports, establish connections to any of the daemons, etc.

As an example let's trace a connection as a user connects via one of the tnc ports.



A) The tncd process talks with the tnc in KISS mode. The AX25 protocol is handled in the tncd process. When a connect request is received the tncd process spawns a BBS process and establishes socket connects between them, data and control.

B) The bbs process knows the callsign of the user and the connection method via it's command line arguments. A connection is established to the bbsd process and a login attempt is issued. The bbsd process will respond with OK or a reject message that should be passed on to the user. This reject message was supplied either by the sysop when the port was locked or by the process that locked the port.

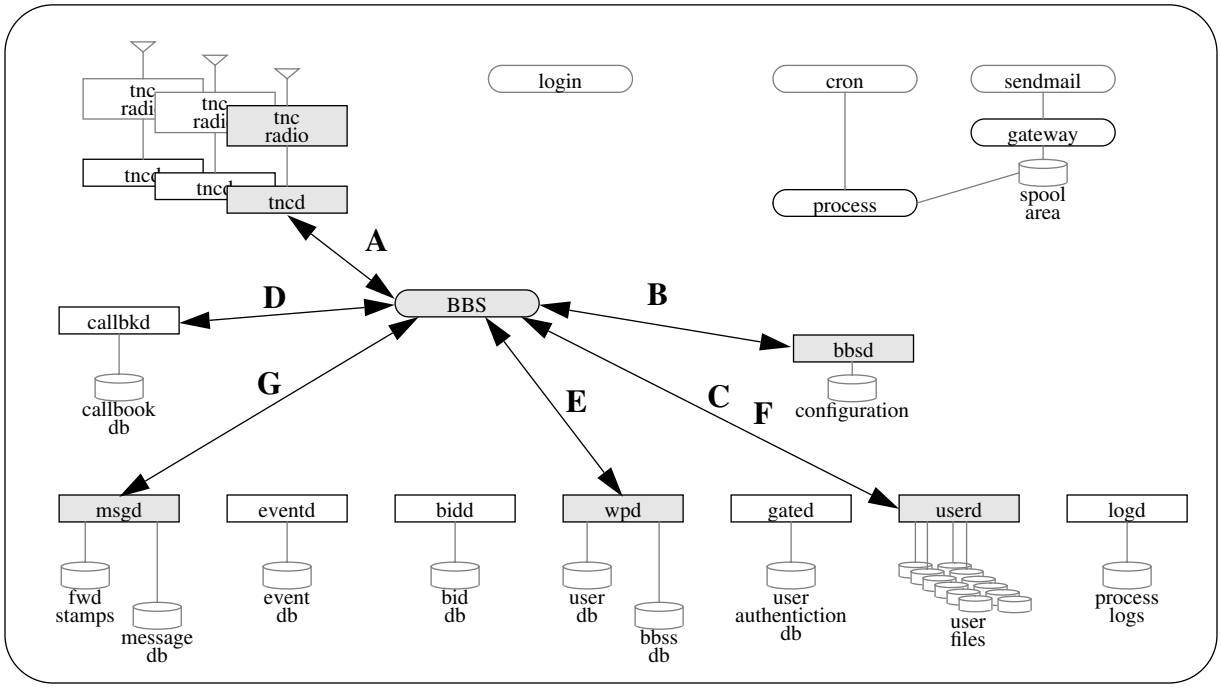
C) If the program is authorized to run it then contacts userd, the user daemon. This process handles the user files resource. Each user has a file that defines his operating environment. The bbs process will ask if the user is ok'd to connect via the port in question. If approved addi-

tional configuration parameters will be uploaded to the bbs process. The bbs process requests a fresh copy of all parameters whenever the main prompt is reached. This keeps the data valid even if the users profile is being changed while he is connected.

D) The bbs will then open a socket to the white pages daemon, wpd, to check on consistency of the white pages data. If some fields are empty, home bbs is incorrect, etc., the user will be prompted for the missing data. At this point the user is fully logged in and his macro 0 is executed.

E) The most common command found in macro 0 is LIST MINE. To fulfill this request the bbs process opens a connection to the message daemon, msgd, and extracts the desired listing. This is then presented to the user and the bbs awaits the users next command by entering a select.

This represents the sequence when an existing user connects to the bbs. When a new user connects it is slightly different.



A) The tncd process spawns bbs as in first example

B) The bbs process gets permission to run from bbsd.

C) An enquiry is made to the user daemon which responds with a user not found.

D) The bbs opens a connection to the callbook daemon, callbkd. If the user is in the callbook then the bbs fills in as many fields as possible.

E) A connection is established to wpd to determine if the user is already know to the bbs network. The data from the callbook and wpd is then used to update the users WP entry.

F) The bbs process then submits a request to userd to create a new user. The user will be prompted for some information not available via the callbook or white pages database. This info will be passed to userd to initialize the user.

G) The default macro 0 for a new user issues the LIST MINE command so we connect to msgd to fill the request.

By using daemons to control the resources of the bbs we avoid data race conditions and the problems associated with locking and unlocking files. All the daemons accept commands in plain text and respond in the same. This allows the sysop to connect directly to the daemons to preform system maintenance and/or test.

From code writers point of view there is benefit as well. The daemons are small programs with a restricted command set and very predictable results. Whenever a change is made to the daemons code a regression test is run against it. This is implemented by a program that runs a script that issues all possible commands to the daemon, alters the date to simulate aging, and issues limited incorrect commands and tests for proper responses. Since the daemons themselves only communicate with other programs, not users, the command set is very restrictive and therefore easy to test.

Conversely the bbs program itself is designed to be incredibly flexible when dealing with the user. It is therefore very hard to test for complete functionality. Making it as small as possible limits the exposure to hidden problems. The same qa process is run on the bbs program but only a small cross section of the command permutations is executed, there are tens of thousands of permutations possible.

Before we address each of the daemons individually let's cover some common points. All the daemons have a common result syntax. If the command was accepted you either get the work "OK" back or the value you were requesting. If an error occurred, improper syntax, etc. the daemons will respond with "NO," followed by the a reason. Simply speaking all responses should be checked for the string "NO," anything else indicates success.

```
[userd-0.01]
n0ary
OK
n6zzz
NO, not a user
```

Commands that result in a variable length list will be terminated by a single period on a line by itself.

The daemons also have a few commands in common:

EXIT

Close connection to daemon. This is really only used by a sysop to kill a telnet session. Programs will typically just close their end of the socket. The daemon will treat this the same as an exit.

SHUTDOWN

Close connection and shutdown daemon. Keep in mind that if the daemon is listed as a DAEMON in the configuration file read by bbsd it will be re-spawned immediately. You can issue the RESPAWN command to bbsd to keep this from occurring.

?

Each of the daemons will display a command list when the help command "?" is issued. The display is terse and is simply to remind the user of command syntax.

STAT

This command is normally only used for debugging purposes. It has a different meaning and display in each of the daemons.

```
[userd-0.01]
stat
usrfile_show()
 0 N6JLH [CLEAN] 3b210
 1 KG6NI [CLEAN] 35b38
 2 KB2EAR [CLEAN] 44ab8
 3 IMPORT [CLEAN] 50458

[wpd-0.01]
stat
bbs count = 4053, image is DIRTY
user count = 25174, image is DIRTY
```

Each of the daemons identifies itself upon connection. The id is in the form of [name-ver]. This field is only used to guarantee that the server and client can talk with each other.

The data files used by all daemons are kept in editable form. This allows the sysop to make global changes by shutting the daemon down, editing it's associated database files with his favorite editor and restarting the daemon. Remember to inhibit the re-spawning of the daemon by bbsd by using the RESPAWN command.

We will now look at each of the daemons in detail.

BBS control process (bbsd)

/bbs/bin/b.bbsd config_file

The bbsd process is normally started via the rc.local file at boot time. On it's command line it expects the path to a configuration file which it will read and store internally. Some of the contents of the configuration file are used directly by bbsd but the majority is simply stored away for dissemination to clients. The data is stored in context free form.

The command **bbs_config** is used to create a template showing all the required fields for the configuration file. It does this by executing all the daemons with a command line option that causes them to dump their requirements and then exit (Similar to ldd for you unix hounds). The file created by **bbs_config** contains comments to aid in initial setup.

Here is an example of a configuration file:

```
####
## BBS Configuration Rules
####
#
# Entries in this file are keyword value pairs. The keyword is
# a string that starts in column 1 and begins with a letter.
# The value field is separated from the keyword by white space.
#
# Values have an associated type. The possible types are:
# STRING, NUMBER, DIRECTORY, FILE and TIME
#
# When specifying a DIRECTORY or FILE if the first character
# is a slash '/' then the path is considered to be absolute.
# If it begins with a character other than a slash the field
# represented by BBS_DIR is prepended. DIRECTORY values should
# not end with a slash.
#
# The TIME values are a number followed by the quantity
# definition, i.e. SECONDS, MINUTES, HOURS, DAYS, WEEKS,
# MONTHS or YEARS.
#
# Comments are lines that begin with a '#'. Comments cannot
# be appended to the end of KEYWORD/VALUE pairs.
#
#

# DAEMON <name>                 <command_line>
# This is the list of daemons that bbsd will spawn and restart if they
# should disappear. Daemons with the name IGNORE will be spawned but
# then forgotten about, they will not be restarted.

DAEMON IGNORE                   /bbs/bin/bbstool
DAEMON TNC0                     /bbs/bintncd TNC0
DAEMON TNC1                     /bbs/bin/tncd TNC1
DAEMON TNC2                     /bbs/bin/tncd TNC2
DAEMON TNC3                     /bbs/bin/tncd TNC3
DAEMON LOGD                     /bbs/bin/logd
DAEMON WPD                      /bbs/bin/wpd
DAEMON BIDD                     /bbs/bin/bidd
DAEMON GATED                    /bbs/bin/gated
DAEMON USERD                    /bbs/bin/userd
DAEMON MSGD                     /bbs/bin/msgd

BBS_DIR                         /u2/bbs/test
```

```
####
## MSGD - Message Daemon
####
#
BBS_CALL                    N6ZFJ
BBS_HOST                    suny
BBSD_PORT                   44510
#
MSGD_PORT                   44518
MSGD_BODY_PATH              message/Body
MSGD_FWD_DIR                message/Forward
MSGD_ROUTE_FILE             etc/Route
MSGD_SYSTEM_FILE            etc/System
MSGD_GROUP_FILE             etc/Group
#
MSGD_AGE_INTERVAL           1 hour
MSGD_AGE_ACTIVE             7 days
MSGD_AGE_KILLED             2 days
MSGD_AGE_OLD                2 days

####
## BIDD - Bid Daemon
####
#
BIDD_PORT                   44514
#
BIDD_FILE                   message/bid.db
BIDD_FLUSH                  2 minutes
BIDD_AGE                    3 months

####
## USERD - User Daemon
####
#
USERD_PORT                  44516
USERD_ACC_PATH              user
USERD_AGE_INTERVAL          1 hour
USERD_AGE_SUSPECT           1 week
USERD_AGE_HOME              4 months
USERD_AGE_NONHAM            1 months
USERD_AGE_NOTHOME           3 months
```

Commands:

LOGIN call via

proc_number

NO, reason

bbsd tracks processes by a process number. This is because a single user can be connected multiple times on different ports. When bbsd gives permission to run it returns the process number for the process making the request. Other uses for this number will be seen in the NOTIFY description.

```
LOGIN N6ZFJ TNC1
145

LOGIN N6ZFJ PHONE0
NO, BBS maintenance in progress
```

PORT number

OK

Each bbs process maintains a socket that can be connected to to allow eavesdropping of the user session. Once connected the sysop can monitor the connection or by typing ON and OFF talk directly to the user. In the case of daemons this is their main port. This command is used to tell bbsd what port a program is listening.

```
PORT 3541
OK
```

CHECK call [via]

cnt

Many processes have to be assured they are the only copy running. This command returns the number of other processes meeting the criteria specified.

```
CHECK N6ZFJ
1

CHECK FWD TNC0
0
```

MSG string

OK

The bbs console shows the users call, how he entered the system, how long he has been connected, the chat port number and a text field. This text field is normally the last command issued when dealing with users, with daemons it is status information. This command is used to upload this text string to bbsd.

```
MSG list all to sale about icom since 10/14/93
OK
```

CHAT ON|OFF

OK

When a user wishes to talk directly to the sysop, keyboard to keyboard, it is reflected in the bbs console display. This command is how the bbs process notifies bbsd and in time the sysop that the user wishes to talk. The sysop then initiates a session to the chat port.

```
CHAT ON
OK

CHAT OFF
OK
```

SHOW variable

value

NO, Variable not found

The majority of fields found in the configuration file read by bbsd are not meant for bbsd. This command is how other processes get access to the info in the configuration file.

```
SHOW MSGD_SYSTEM_FILE
/u2/bbs/test/etc/System

SHOW MSGD_DUMMY_FILE
NO, Variable not found
```

SET variable value

OK

This command allows a variable to be redefined dynamically. It is mostly for debugging purposes and altering AX25 parameters.

```
SET TNC0D_MAXFRAME 3
OK
SET TNC0D_PACLEN 224
```

SHOWORIG variable

value

NO, Variable not found

This command is identical to SHOW except it will show the original contents of the variable prior to it being changed by SET.

```
SHOW BBS_FWDCALL
N6ZFJ-2
SET BBS_FWDCALL N6ZFJ-4
OK
SHOW BBS_FWDCALL
N6ZFJ-4
SHOWORIG BBS_FWDCALL
N6ZFJ-2
```

LOCK via reason

OK

Lock the port for this reason

```
LOCK PHONE1 Please use other phone line
```

UNLOCK via

OK

Unlock the port

```
UNLOCK PHONE1
```

NOTIFY ON|OFF

OK

One class of programs that connects to bbsd is the status type, i.e. the bbs console. In normal operation bbsd will not talk to a user unless it is asked a question. This command will force bbsd to dump status information whenever a state changes. The data is different from standard responses in that it starts with a '!'.

!LOGIN # call via login_time

!LOGOUT #

!PORT # number

!MSG # string

!CHAT # ON|OFF

!LOCK via string

!UNLOCK via

The '#' that follows most of the commands is the process number.

STATUS PROCS

proc_num call via port login_time idle_time chat message
LOCK via reason
UNLOCK via

Prior to going into NOTIFY ON mode the status process will want to know the current condition of all active processes. This command dumps this information.

```
13 N6UNE TNC0 3476 753207686 753207740 OFF sp n6zfj
15 N6UVY TNC0 3484 753207696 753207752 OFF list mine
16 N0ARY CONSOLE 4501 753207900 753207920 OFF list held | read why
TNC0 UNLOCK
TNC1 UNLOCK
TNC2 UNLOCK
PHONE0 UNLOCK
PHONE1 LOCK Please use other phone line
CONSOLE UNLOCK
SERVER UNLOCK
DAEMON UNLOCK
SMTP UNLOCK
STATUS UNLOCK
```

STATUS LOCK

LOCK via reason
UNLOCK via

Same as STATUS PROCS but only dumps the condition of the locks.

```
TNC0 UNLOCK
TNC1 UNLOCK
TNC2 UNLOCK
PHONE0 UNLOCK
PHONE1 LOCK Please use other phone line
CONSOLE UNLOCK
SERVER UNLOCK
DAEMON UNLOCK
SMTP UNLOCK
STATUS UNLOCK
```

RESPAWN daemon_name [ON|OFF]

OK

This command is issued if the sysop wishes to keep bbsd from re-spawning a daemon or re-enabling the spawning. If the sysop wishes to work on a daemons database he must first set RESPAWN OFF, issue a SHUTDOWN command to the daemon in question, edit the daemons database, then set RESPWAN back to ON. Not supplying the ON or OFF operand will display the current status of the re-spawn flag for that daemon.

```
RESPAWN BIDD OFF
OK

RESPAWN BIDD
OFF

RESPAWN BIDD ON
OK
```

User accounts (userd)

/bbs/bin/b.userd [-w|W] [-f] [-l|L] [-p bbsd_port] [-h bbsd_host] [-d#]

-w	**Write configuration dependencies**
-W	**Write configuration dependencies with opening comment**
-f filename	**Append configuration dependencies to filename, requires -w or -W**
-l	**Enable daemon logging, keep everything (debugging)**
-L	**Enable daemon logging, clear log file when daemon is idle (debugging)**
-p bbsd_port	**Where to find bbsd if different form the compiled in port**
-h bbsd_host	**Where to find bbsd if different from the compiled in host**
-d #	**Debugging flags**

The userd process is responsible for maintaining the user account files. Each user has a file that defines the individual parameters that control how the bbs deals with a user. It is also responsible for aging user accounts.

This is an example of a user account file:

```
+0 1 753346046 5 1753
NUMBER 0
LNAME Arasmith
PHONE 408-749-0501
ADDRESS bob@arasmith.com
FREQ 443.775+/224.28-/145.62s all 100hz PL (LERA's Linked System)
TNC AEA
SOFTWARE BBS
COMPUTER Sun
RIG Yaesu
LINES 50
INCLUDE >BAYCOM
INCLUDE >LERA
MESSAGE 751917277
SYSOP
APPROVED
EMAIL
LOGGING
IMMUNE
REGEXP
ASCEND
HELP 2
MACRO 1 read mine; kill mine
MACRO 2 list unread club
MACRO 8 me
PORT TNC0 OK 2 736062604 737667313
PORT TNC1 OK 0 0 0
PORT TNC2 OK 3 730189175 730191787
PORT PHONE0 OK 18 729112068 751002422
PORT PHONE1 OK 0 0 0
PORT CONSOLE OK 1733 728666739 753346046
PORT SERVER NO 0 0 0
PORT DAEMON NO 0 0 0
PORT SMTP OK 1016 738999699 753319429
PORT STATUS OK 0 0 0
PORT TNC3 OK 1 738384730 738384730
```

When userd is started it scans the user accounts directory and reads the first line from each file. This line tells userd when the last time that user connected, via what method, how many times the user has connected, etc. The remainder of the file is ignored at this time.

When a user connects or another user issues the command to view the users file userd will re-read the file and present it to the requesting process. No other process besides userd knows of the user account files.

The user account files don't contain all the parameters to define a user. The file only contains those parameters that differ from the default set. Each of the user parameters will be described briefly with the commands to access/modify them.

Userd Control Commands:

These command control the operation of userd itself. They are independent of the user accounts themselves.

call

OK

NO, not a user

This command is used to focus userd on a specific user. If the user is valid the users account file is loaded into memory and held for the duration of the connection. Even if additional users are subsequently the focus. This allow quick access to previous users. If two processes are focused on the same user they share the same image in memory. The user will not be purged until all dependencies on the user are gone.

```
N6ZFJ
OK

N6ZZZ
NO, not a user
```

AGE

OK

Initiate an aging pass over the user accounts. Aging is normally ran automatically at the interval defined in the configuration file as USERD_AGE_INTERVAL, normally 1 hour. Users are purged from the system when there have not logged in directly or sent a message through the internet gateway in the specified time. The time varies by the type of account, those claiming the bbs as home are given longer idle times than those that have homes elsewhere. Those that are non-hams are purged quicker than hams. All of these parameters are defined in the configuration file.

CALL

call

NO, must set call first

This is obviously a sysop command. It is used to see what user currently has focus. A program wouldn't forget :)

```
CALL
N0ARY
```

FLUSH

OK

It is often the case where you know you will not be going back to a user again. You therefore do not want to tie up memory in the daemon. The FLUSH command is used to purge all users currently attached to your process. It doesn't disturb any users currently attached to other processes.

KILL

OK

This command deletes the users accounts file and flushes all reference to him in the internal lists. There is no going back.

CREATE call

OK

Creates the user entry in the internal lists, and initializes the user's account file. The default settings are located in the configuration file.

USERD_DEF_MACRO_0	list mine; list unread clubs
USERD_DEF_MACRO_1	read mine; kill mine
USERD_DEF_MACRO_2	list unread club
USERD_DEF_MACRO_3	list unread from n0ary to users | read
USERD_DEF_MACRO_4	users 20
USERD_DEF_MACRO_5	ports
USERD_DEF_MACRO_6	info manual
USERD_DEF_MACRO_7	info commands
USERD_DEF_MACRO_8	info shell
USERD_DEF_MACRO_9	"help all"
USERD_DEF_LINES	20
USERD_DEF_ALLOWED	TNC0 TNC1 PHONE0 CONSOLE SMTP STATUS
USERD_DEF_HELP	3
USERD_DEF_LOGGING	TRUE
USERD_DEF_APPROVED	TRUE
USERD_DEF_LISTASCEND	FALSE

LOCATE usernumber

call

NO, expected a user number

NO, user by that number

Each user of the system has a unique number that identifies that user. For the most part this is not used but one of the features of the bbs is a hook to a DecTalk speech synthesizer and a dtmf decoder. The setup allows users to ask the bbs questions via their voice radio to see if it is necessary to login. They can see how many messages are waiting for them, who they are from, etc. They access these functions via their user number.

```
LOCATE 7
K6GSJ
LOCATE N0ARY
NO, user by that number
LOCATE
NO, expected a user number
```

LOGIN via

OK

NO, not authorized for access via this port

This command is used to register the user as connecting to the bbs. If the user is allowed to connect via this method OK is returned, otherwise NO. The users account is timestamped with the connect time and method.

SHOW

Show is a sysop command to dump the current memory image of the user. In standard operation programs will ask for the particular field they are interested in and not ask for them all at once. The format of the output is very similar to the format of the user account file shown above.

USER call|count

user entry

user list

NO, expected callsign or count

This command displays either the last time a specific user connected or the last X number of user to connect.

```
USER 6
KE6BWH 4 TNC0 753344161
IMPORT 11497 SMTP 753343830
KD6MZM 523 TNC0 753343729
WN6U 739 PHONE0 753343291
KD6ZXT 9 TNC0 753343006
K6TAM 58 TNC0 753342967

USER N0ARY
N0ARY 2771 CONSOLE 753348582

USER
NO, expected callsign or count
```

User Account Commands:

There are basically four types of variables associated with user accounts, toggles, strings, lists and numbers. Since userd is often used by the sysop directly the commands are a little more flexible than the other daemons. We will show the different methods of editing and displaying each field here as generic operations then go through the variables themselves.

Toggles:

variable

ON

OFF

Display the state of the variable.

SET variable
variable ON

OK

Set variable to on.

CLEAR variable
variable OFF

OK

Set variable to off.

Strings:

variable

Show the current contents of the variable.

variable string

OK

From the first non-blank character after the variable name to the end of the line is copied into the variable. White spaces inside the string are allowed.

CLEAR variable

OK

Null out the variable.

Numbers:

variable

Show the current contents of the variable.

variable number

OK

NO, expected a number operand

Set variable to a new number.

Lists:

variable

result_list

The multi line result is displayed. The list is terminated by a period on a line by itself.

variable string

OK

Add the supplied string to the list associated with the variable. Note that this is a string, if the string contains multiple words they will not each get their own list location.

CLEAR variable

OK

This clears the entire list at once. Deleting a single item out of the list has to be handled by the client process.

User Account Variables:

Some of these variables are not intended to be under the control of the user. There is no protection built into the daemon to distinguish between the two types of commands. It is assumed that the client processes will provide this defense. A sysop connecting directly to the daemon has full control over all fields.

Unlike the White Pages entry for a user, none of the user account information is sent off the bbs. It is for local consumption only.

User Info:

All of these fields are general purpose. If they are loaded with garbage data it will have no affect on the bbs process itself. They are purely informational.

COMPUTER **[string]**
RIG **[string]**
SOFTWARE **[string]**
TNC **[string]**

Users can voluntarily supply information as to their packet setup. This information is used to put new users in contact with experienced users that may share their same configurations.

COUNT **[read only field]**

Display the number of times a user has connected to the bbs.

FREQ **[string]**

The user is asked to load this field with voice frequencies where he frequently monitors. Just an aid to helping track down a packet person.

LNAME **[string]**

This is the users last name.

PHONE **[string]**

Users phone number

Interaction Variables:

ADDRESS email_address **[string]**

This is the email address for a distant network where the user can be reached. Setting this variable does not enable the automatic forwarding of mail, see EMAIL.

APPROVED **[toggle]**

Users that are not APPROVED have all mail held for review before forwarding to another bbs.

BBS **[toggle]**

This variable gives the client a hint as to the fact that this user may be a bbs. The messages-of-the-day is suppressed and all prompts are minimal. The reception of the SID from the other system confirms the BBS status.

EMAIL **[toggle]**

OK

NO, must set email ADDRESS first

This variable in conjunction with the ADDRESS variable control the automatic forwarding of packet mail to a users internet/compuserve/mcimail/etc. account. The ADDRESS command must be set prior to setting this field. Otherwise an error message will be generated. When a message is forwarded to a distant network the message is marked as read on the bbs and killed.

HDUPLEX **[toggle]**

This variable only applies to connections made via telephone modem. It can be set or cleared as desired by the user. In half duplex mode the bbs will not echo characters back to the user.

HELP **[number]**

The bbs has 4 help levels (0,1,2,3). The default for new users should be level 3. This level has very verbose error messages, will prompt a user through commands to get adequate info, etc. Level 2 has fairly verbose error messages but will normally not prompt the user for additional information. Level 1 is for the expert user. The error messages are terse and the bbs just does what it is told, even if it is wrong. Level 0 is for bbs to bbs connections only. There is no prompting and illegal commands will terminate the link.

IMMUNE **[toggle]**

If a users has this variable set he user account is immune to aging and all of his messages will be immune to aging. This variable can be set by the sysop or via the VACATION command on the bbs. BBSs should have their IMMUNE variable set to make sure they are not purged.

LINES **[number]**

This variable represents the number of lines the users screen is capable of displaying. When the bbs as displayed this number of lines it will pause and prompt the user for an input. The user can then command the bbs to display the next page of data, abort the listing, scroll to the end without additional pauses and/or preload a command to be executed when the listing is complete.

LOG **[toggle]**

If this variable is set then all keystrokes to and from a user are logged into a file unique to the user.

MACRO # **[string]**

Each user has 10 keystroke macros in which to load commands or sequences of commands. Each macro can hold up to 80 characters of commands, multiple commands in the same macro must be separated by semicolons. The macros can call other macros to give more than 80 characters worth of commands. They can also be nested. Macro 0 is special in that it is run automatically at login, similar to a .cshrc file in unix or autoexec.bat file in dos. To execute a macro the user simply types the macro number and returns.

MESSAGE **[number]**

This is a time stamp that represents the last message the user listed with the LIST NEW command.

NEWLINE **[toggle]**

To save screen space and make the display more pleasurable to read the bbs will prompt a user then wait at the end of the line for a response. This puts the users response on the same line as the prompt. Some packet software will not display a line to the user until the carriage return is received. If this variable is set then all prompts will be followed by a carriage return.

NONHAM **[toggle]**

Identifies the user as a non-ham. More than likely gained access via the phone ports. Non-hams have very restrictive access to the bbs. They can only send secure messages for instance.

NUMBER **[read only field]**

This is the user number. Really only need to use the voice synthesizer port. Numbers are assigned automatically by userd when a user is created.

PORT via **[toggle]**

The bbs has multiple methods to gain access. Each method (port) is under the control of the sysop. Users can be restricted from entering on bbs to bbs channel, restricted from the SMTP email gateway, etc.

REGEXP **[toggle]**

Those familiar with regular expression syntax can choose to have it used in all search criteria. The default is off.

SIGNATURE **[toggle]**

Users can create custom signatures and have them appended to messages on demand or automatically. This variable indicates whether it is automatic or not. The signatures themselves are located external to the users account. The functionality is similar to the unix .signature file.

SYSOP **[toggle]**

User is allowed to elevate to sysop status. This still requires a password to be supplied and can only be done from a console or phone connection.

VACATION **[toggle]**

Similar to the unix vacation function. It allows a user to specify a message that will be mailed in receipt of message. In addition to this automatic replying the users messages are immune to aging as well as his account.

Message Listing Variables:

ASCEND **[toggle]**

This defines the order in which message are presented to the user, newest to oldest or oldest to newest.

EXCLUDE **[list]**

One of the main complaint from users is all the garbage on packet. The problem is one persons garbage is someone else's treasure. This command and it counterpart INCLUDE can be used to control what messages are seen by the user. The EXCLUDE list is a filter that is tested against every message as the listing is being created.

```
EXCLUDE all@allus
EXCLUDE sale@allus
EXCLUDE keps
EXCLUDE @amsat
```

INCLUDE **[list]**

This command is the counterpart to EXCLUDE. Where EXCLUDE identifies filters of what messages should be hidden, This list identifies filters that are used to draw attention to messages that match. The bbs command LIST CLUBS is used to display message matching this criteria.

Configuration Variables:

USERD_PORT	**44516**
USERD_ACC_PATH	**user**
USERD_AGE_INTERVAL	**1 hour**
USERD_AGE_SUSPECT	**1 week**
USERD_AGE_HOME	**4 months**
USERD_AGE_NONHAM	**1 months**
USERD_AGE_NOTHOME	**3 months**

Message Database Management (msgd)

/bbs/bin/b.msgd [-w|W] [-f] [-l|L] [-p bbsd_port] [-h bbsd_host] [-d#]

-w	**Write configuration dependencies**
-W	**Write configuration dependencies with opening comment**
-f filename	**Append configuration dependencies to filename, requires -w or -W**
-l	**Enable daemon logging, keep everything (debugging)**
-L	**Enable daemon logging, clear log file when daemon is idle (debugging)**
-p bbsd_port	**Where to find bbsd if different form the compiled in port**
-h bbsd_host	**Where to find bbsd if different from the compiled in host**
-d #	**Debugging flags**

The msgd process manages the message database and presents the information to the user processes. It also handles setting up forwarding, putting bulletins into a newsgroup form and message aging. There are two directories used by msgd, one for storing the messages themselves and the other is for stamps indicating messages awaiting forwarding.

Messages are kept in files named after their message number on the system. The file contains the message body terminated with a "/EX" followed by RFC-822 fields that describe all characteristics of the message. Here is an example of a live message.

```
R:931114/1920 @:N6IIU.#NOCAL.CA.USA.NA Palo Alto #:1130 Z:94301
R:931115/0258 9763@N0ARY.#NOCAL.CA.USA.NA Sunnyvale Z:94086
R:931111/2009 48858@N6QMY.#NOCAL.CA.USA.NA Fremont Z:94536
R:931111/1956 6368@N6IYA.#NOCAL.CA.USA.NA
R:931111/1958 5523@KA6FUB.#NOCAL.CA.USA.NA
R:931111/1909 16339@KI6YK.#NOCAL.CA.USA.NA
R:931111/1755 3410@WA6YHJ.#NOCAL.CA.USA.NA

WUD LIKE TO PKT WIT HAMS IN ROCHESTER, N.Y., AKRON,
OH., TUCSON, AZ., ES HEMET, CA. ALSO N-E ONE IN THE
841 SIG. CORPS FROM 1942-1945. TNX. LEN IN PLEASANTON,
CA. BBS==WA6YHJ-1.#NOCA.CA.
/EX
From: N6TBA@WA6YHJ
To: WANTED@ALLUSA
Subject: LOOKING FOR HAMS IN SEVERAL CITIES****
X-Bid: 3410_WA6YHJ
X-Create: 753334097
X-Type: B
X-ReadBy: KC6WPG
X-ReadBy: N6JLH
X-ReadBy: K6SDE
```

This is a list of the RFC-822 fields and their definitions that can occur in a message.

To: call@home[.hloc]
From: call@home[.hloc]
Subject:
X-Bid:
X-Type: P|B|T|S
X-Password:

> These are the only fields necessary to maintain a message and forward it to other bbss. The X-Bid field is optional and is only required on bulletins. A message type of "S" represents a SECURE message. Secure messages can only be read over a secure link, telephone or internet gateway. Secure messages can also carry passwords.

X-Create: time_t
X-Old: time_t
X-Kill: time_t

> These fields imply the current status of the message. When a message is first recived it is stamped with an X-Create entry containing the time_t value representing the date and time of the creation. As the message is aged or killed by a user other stamps are appended, X-Old or X-Kill. Each of these carries the date/time of the operation. As the RFC-822 fields are parsed the last field encountered is the status of the message. If a killed message is re-activiated an X-Create entry is simply appended to the message.

X-Immune: ON

> A message that has the last status field of X-Immune is immune to aging. The immune setting can be set to off by activating the message, appending a X-Create field, or by killing the message, appending a X-Kill field.

X-HeldBy:
X-HeldReason:

> Messages can be held by users, sysops or by the bbs itself as a result of problems with routing or translation. When anyone holds a message the bbs will prompt the user for a reason. The call of the user and the reason for the hold are kept in the message. To cancel a hold the sysop need only activate the message or kill it.

X-ReadBy:

> The bbs keeps track of the calls of everyone that has read the message. This allows users to utilize the keywords of UNREAD and READ to the LIST commands. It also allows the user to see who has read a message by using the READ WHO command.

Msgd never rewrites a message file it simply appends RFC-822 entries. If two To: fields are found it is the last one encountered that takes precedence. This is the case with all the entries with the exception of X-ReadBy and X-HeldReason, which are maintained as lists.

The first implementation of the bbs had a message directory file that was a shared resource for all running bbs programs. They would use file locking to make sure the data was kept in tact and two processes didn't clobber each other with overlapping writes. Each bbs process would check the time stamp on the file to see if it had been modified since the last time it was read. This became very costly timewise. The goal of the msgd concept was to limit the amount of overhead associated with building these message lists and maintaining them inside each of the bbs processes.

The sequence of events is that the user process connects to msgd and identifies the user by call. Msgd can then build a message list that represents just the messages the this user is allowed to see. As soon as the user process requests a LIST the entire list is dumped to the user process. It is then responsible for filtering it further for the user via the command line operands or the INCLUDE/EXCLUDE lists in the users account. As other processes interact with msgd the message list is altered, messages are killed, sent, read, etc. When the original user process makes another LIST request rather than getting the entire list again it is supplied with just the diffs from the previous list. The user process can request the entire list be dumped again by preceeding the LIST with a FLUSH command.

This bbs also will put messages into a newsgroup type of format. The MSGD_GROUP_FILE is used to define what type of bulletins should be placed in a particular group. Here is an example of that file:

```
>TCP:          TCPIP
>NOS:          TCPIP

>PC:           COMPUTER IBM
>DOS:          COMPUTER IBM
>MSDOS:        COMPUTER IBM
>WINDOWS:      COMPUTER IBM
>C64:          COMPUTER
>COCO:         COMPUTER
>APPLE:        COMPUTER APPLE
>MAC:          COMPUTER APPLE
>AMIGA:        COMPUTER AMIGA
>VIRUS:        COMPUTER IBM
```

A single message can exist in multiple groups. A user can display the available groups and the number of messages currently in each group with the GROUP command. When a user selects a group msgd will only display messages that are in that group. Personal and NTS messages will be hidden as well as bulletins that are not in the group. Users can jump between groups by just issuing another group command and leave the group mode entirely by typing GROUP OFF.

Commands:

The following commands are configuration to inform msgd of how you wish to have data presented, etc.

USER call

OK

This is the first command that should be issued to msgd upon connection. It identifies who you are and defines a focus for building of the initial message list.

SYSOP
NORMAL

OK

Upon connection the user is in NORMAL mode. This means that the internal representation of his messages is focused for him. He will not be able to see personal message addressed to others and messages that are not normally seen by users. By issuing the SYSOP command all restrictions are removed and the entire message list will be displayed. You can go back and forth between this states at any time. There is a delay involved with the first LIST command following a state change. A FLUSH command is issued whenever a state change occurs which means that the internal message list has to be recreated. The bbs program is responsible for authentication of the user prior to issuing the SYSOP command.

DISP BINARY|NORMAL|VERBOSE

OK

NO, Expected VERBOSE|NORMAL|BINARY

Msgd has three forms that it can use to output it's data. The VERBOSE form is the form that is human readable and pleasing to the eye. The problem is that it isn't very computer friendly. So if the sysop connects directly to msgd he should use the DISP VERBOSE command.

```
46317 B OM 1709 WANTED@ALLUS WB1ELA 1 0926/0441 ACC FOR FT-203R
46325 B K 1097 WANT@USBBS KF2GC 0 0926/0521 MODS FOR IC-W21AT & IC-737
46362 B K 1650 USERS@N6QMY N6ZFJ 25 0926/0811 Re: [!cmd]
46390 B K 1133 SALE@ALLUS W1LUS 0 0926/1221 BOOKS FOR C-64
46391 B KM 4115 BAYCOM@ALLUS WB3FKP 5 0926/1221 BAYCOM SOFTWARE
```

The NORMAL setting is still fairly user friendly but the dates are in time_t values as opposed to the 931103/1245 form we are used to dealing with. The display is also not formatted so the columns line up. But the bbs program can easily parse the data.

```
46317 1709 132 $7182_WA1PHY WANTED@ALLUS WB1ELA 749043665 1 ACC FOR FT-203R
46325 1097 132 $47316_KD2AJ WANT@USBBS KF2GC 749046098 0 MODS FOR IC-W21AT & IC-737
46362 1650 2188 $ USERS@N6QMY N6ZFJ 749056264 25 Re: [!cmd]
46390 1133 148 $7126_WA1PHY SALE@ALLUS W1LUS 749071265 0 BOOKS FOR C-64
46391 4115 132 $18172_KB3QW BAYCOM@ALLUS WB3FKP 749071272 5 BAYCOM SOFTWARE
```

The last form is BINARY. This bbs supports a concept referred to as the Client/Server bbs model. This new generation bbs is still under development but in short it off loads most of the user interface and interactive tasks to the users pc. The interface to the bbs is binary, fast and efficient in comparison to normal operation. This BINARY setting is designed to dump straight to a client.

ACTIVE msgnum
OLD msgnum
KILL msgnum
KILLH msgnum
IMMUNE msgnum

OK

NO, Message not found

These commands alter the status of a message. The only one that is not reversible is the KILLH (KILL HARD) command. This command removes the message body file from the file system. All of the other commands simply append the appropriate RFC-822 entry to the message file and set a couple bits in the messages entry in the internal list.

HOLD msgnum

OK

NO, Message not found

NO, Trouble reading

This command alters the status of the message just like the above commands but it is different in that it expects to receive a list of that it will append to the message body as X-HeldReason entries. It accepts all lines up to a line that contains only a period.

PARSE msgnum

OK

NO, Message not found

NO, Error while parsing RFC822 fields

If a message has a serious problem the sysop can go in and manually edit the RFC-822 entries in the message itself. He can then issues this command to msgd to have it rebuild the message entry in the internal list by re-parsing the message file.

LIST

msg_list

The actual format of this list is dependent on the DISPLAY setting as discussed above. For the sake of readability the examples will be in VERBOSE mode. The first listing requested by a user will be the entire message list, or the portion that he is allowed to view.

```
LIST
46317 B OM 1709 WANTED@ALLUS   WB1ELA      1  0926/0441 ACC FOR FT-203R
46325 B K    1097      WANT@USBBS   KF2GC       0  0926/0521 MODS FOR IC-W21AT & IC-737
46362 B K    1650    USERS@N6QMY   N6ZFJ      25  0926/0811 Re: [!cmd]
46390 B K    1133      SALE@ALLUS    W1LUS       0 0926/1221  BOOKS FOR C-64
46391 B KM 4115 BAYCOM@ALLUS    WB3FKP      5 0926/1221  BAYCOM SOFTWARE
.
```

Now we assume another user has read message 46325 and message 46390 is deleted from the system and a new message was just received. Our next list command will only show us the differences.

```
LIST
* 46325 B K 1097 WANT@USBBS KF2GC 0 0926/0521 MODS FOR IC-W21AT & IC-737
- 46390
+ 46401 B K 1290 SALE@SBAY  N6UNE   0 0926/0703 ICOM W2A
.
```

The lines that are altered begin with a ‘*’, the lines that are new are represented by ‘+’ and messages that should be removed from the list have a ‘-’. The system makes updates to the bbs’s copy of the message list very fast.

READ msgnum

READH msgnum
READRFC msgnum

These commands will dump the message body to the user process. If the READH (READ HEADER) command is issued the routing headers will be supplied as well. The READRFC command doesn't dump the message body but just the RFC-822 entries associated with it.

WHO msgnum

This command dumps a list of all the users by callsign that have read the message supplied. The list is terminated by a single period.

WHY msgnum

Dump who has held the message and why to the user process.

SEND

OK, msgnum

The user process is responsible for putting the message into the proper form to be written directly into the message file. The message should begin with routing headers if they exist, followed by message body terminated with a "/EX". Then all the RFC-822 entries that define the message.

Upon successful transfer msgd will respond with the message number assigned to the message.

```
SEND
This is a test to see if a message can be sent. This
was generated as a local user would. Notice the absence
of routing headers.

73, bob
/EX
From: N0ARY
To: WA6BXH@N0ARY.#NOCAL.CA.USA.NA
Subject: Hello Again!
X-Create: 749044382
X-Type: P
.
OK, 4376
```

FLUSH

OK

This command is used to force msgd to re-initialize the internal message list for the user process. The next LIST command will be a full listing rather than deltas.

EDIT msgnum rfc822

OK
NO, Message not found
NO, Could not parse RFC822 field
NO, Not an editable field

This is the mechanism for editing different fields of a message. A valid RFC-822 entry is sent to msgd to be appended to the message file. The message is then re-parsed and the internal message list updated.

ROUTE msgnum
ROUTE address

These commands will show the user process where a specific message is to be routed or where an address will be routed.

```
ROUTE 46370
FwdOn: AA4RE
N6QMY
.

ROUTE ALLUS
FwdOn: ALLUS
N6QMY
K3MC
N6LDL
N6IIU
KB6MER
W2XO
KB2EAR
.
```

PENDING
PENDING bbscall
PENDING msgnum

These commands are used to determine what messages are waiting to be forwarded, how many messages are waiting to go to a specific bbs or where a particular message is be forwarded.

```
PENDING
B.43765.N6QMY
B.43765.N6LDL
P.44001.KB6MER
P.44002.N6QMY
.

PENDING N6QMY
43765
44002
.

PENDING 43765
N6QMY
N6LDL
.
```

FORWARD msgnum [bbscall]
OK

This command is to tell msgd that a message has been successfully forwarded to a bbs. Upon receiving this command msgd will delete the message stamp from the forwarding directory. This command can also be used to cancel the forwarding of a message. You simply leave off the bbscall and all forwarding references to the file will be deleted.

GROUP
GROUP name
GROUP OFF

These commands are used to access the GROUP mode of displaying messages. The GROUP command with no options will display the available groups and the message count in each group. By supplying a name the listing focus is moved to just that group. To leave group mode and return to normal listing functionality the OFF operand is used.

```
GROUP
WANT 5
SALE 10
SPACE 0
MISC 5
HELP 2
JUNK 3
AMSAT 2
PACKET 1
.
GROUP SALE
OK
LIST
46322 B 796 SALE@USBBS WA2OLZ 4 0926/0521 1GB SCSI disk - sale/swap
46324 B 957 SALE@USBBS N2SXX 0 0926/0521 EPSON HX-20 computer cheap
46327 B 1259 SALE@USBBS N2JBJ 0 0926/0521 COLOR COMPUTER II SYSTEM $1
46330 B H 1019 SALE@USBBS N1DBE 0 0926/0522 Kenwood TM-431 440 Mobile Ri
46333 B 1786 SALE@USBBS AA1AX 6 0926/0522 FT1000 Yeasu
46342 B 680 SALE@USBBS WA2MCZ 2 0926/0524 kenwood TL-922A
.

GROUP WANT
OK
LIST
46317 B M 1709 WANTED@ALLUS WB1ELA 1 0926/0441 ACCESSORIES FOR FT-203R HAND
46325 B 1097 WANT@USBBS KF2GC 0 0926/0521 MODS FOR IC-W21AT & IC-737
46329 B 535 WANT@USBBS WA2QWM 0 0926/0521 WANTED 6 METER MODULE YAESU
46330 B H 1019 WANT@USBBS N1DBE 0 0926/0522 Kenwood TM-431 440 Mobile Ri
46344 B 969 WANT@USBBS N2FYS 0 0926/0524 IC-730 100W PA MODULE
.

GROUP OFF
OK
```

Configuration Variables:

MSGD_PORT **44518**

MSGD_BODY_PATH **message/Body**
MSGD_FWD_DIR **message/Forward**

These are the directories where msgd keeps the message files and the forwarding stamps.

MSGD_ROUTE_FILE **etc/Route**
MSGD_SYSTEM_FILE **etc/System**
MSGD_GROUP_FILE **etc/Group**

These are the files used by msgd. They are all opened read only and are closed unless they are actively being read.

MSGD_AGE_INTERVAL **1 hour**

MSGD_AGE_P_ACTIVE **14 days**
MSGD_AGE_P_OLD **2 days**
MSGD_AGE_P_KILLED **3 days**

MSGD_AGE_B_ACTIVE **7 days**
MSGD_AGE_B_OLD **2 days**
MSGD_AGE_B_KILLED **1 days**

MSGD_AGE_T_ACTIVE **99 days**
MSGD_AGE_T_OLD **1 days**
MSGD_AGE_T_KILLED **3 days**

Aging is run automatically every MSGD_AGE_INTERVAL. Messages exists in one of 3 states, ACTIVE, OLD and KILLED. Assuming the above settings a personal message will remain active on the system for 14 days unless it is killed by someone before that. The only thing different about he OLD state is that there is an “O” displayed in the status field of the listing. It is just a warning to the user that the message is about to be killed. After 2 days as OLD the message is changed to KILLED. At this time the message is no longer readable by the user but he can have a sysop recover the message during that 3 day period (see ACTIVE command). After 3 days as KILLED the message file is deleted and the message entry purged from the list.

Different age values can be placed on bulletins, NTS and personal messages. Secure messages are aged as personals.

White Pages (wpd)

/bbs/bin/b.wpd [-w|W] [-f] [-l|L] [-p bbsd_port] [-h bbsd_host] [-d#]

-w	**Write configuration dependencies**
-W	**Write configuration dependencies with opening comment**
-f filename	**Append configuration dependencies to filename, requires -w or -W**
-l	**Enable daemon logging, keep everything (debugging)**
-L	**Enable daemon logging, clear log file when daemon is idle (debugging)**
-p bbsd_port	**Where to find bbsd if different form the compiled in port**
-h bbsd_host	**Where to find bbsd if different from the compiled in host**
-d #	**Debugging flags**

The white pages daemon has two main duties, first to service the bbs processes and secondarily to generate updates to sent to the regional servers. This bbs treats the white pages database slightly different than the other bbs programs. Most bbs programs have a single file containing user call, home bbs, hloc of bbs, users first name, zip and qth. Since this bbs relies heavily on accurate wp info to do addressing and error checking we would prefer not to place our faith in the hloc info received via the wp updates. This bbs scans the message headers for valid hlocs for all the bbss and keeps this info in a separate file. The user file contains the same information as normal with the exception of home bbs hloc, which is kept in another file.

There is one known problem with this system. There is a bbs in Arizona that has both a packet bbs and an aplink bbs. The aplink bbs uses the same call as the packet bbs but adds #APL to the hloc. This gives us two possible hlocs for the same home bbs call. Since this is so rare and is easily handled by having the users force the address in the rare occasions it comes up we will keep the system the way it is. The benefits far outweigh the drawbacks.

The user database looks like this:

```
# v1 /bbs/wp/user.db
#
# This is a machine created file. If you edit it manually
# you need to kill the wpd process, edit the file, then
# restart the wpd daemon.
#
# This file is automatically updated once an hour if changes
# have been made to the runtime memory image.
#
# The '+' character at the beginning of the line indicates
# that the line has been preparsed so error detection can
# be skipped. If you edit or add a line delete the '+'.
#
# Entry format, non-parsed..
# CALL LEVEL FIRSTSEEN SEEN CHANGED UPDATED HOMEBBS FNAME ZIP QTH
# | | | | |
# | +------+------+-------+- mo/da/yr (02/28/93)
# +- 0 = Sysop
# +- 1 = User
# +- 2 = Questionable
#
# N6ZFJ 1 12/14/91 02/01/93 01/15/93 01/01/85 N0ARY Connie 94086 Sunnyvale, CA
#
+KC6RSG 2 0 0 692150400 748346710 W6CUS Andy 94805 Richmond
+KC6RRP 2 0 0 695174400 748346710 N6REB Konrad 95380 Turlock
+IW5DAX 2 0 0 751917392 751957347 I5SGG ? ? ?
+VK3DAW 3 0 0 729216000 748346710 VK3FRS ? ? ?
```

```
+KC6RTA 2 0 0 694915200 748346710 W6PW Dan 94501 Alameda, CA
+KC6TBQ 2 0 0 710640000 748346710 W6PW Gary 94501 Alameda, CA
```

The bbs database looks like this:

```
# v1 /bbs/wp/bbs.db
#
# This is a machine created file. If you edit it manually
# you need to kill the wpd process, edit the file, then
# restart the wpd daemon.
#
# This file is automatically updated once an hour if changes
# have been made to the runtime memory image.
#
# The '+' character at the beginning of the line indicates
# that the line has been preparsed so error detection can
# be skipped. If you edit or add a line delete the '+'.
#
# Entry format, non-parsed..
# BBSCALL LEVEL HLOC
# |
# +- 0 = Sysop
# +- 1 = User
# +- 2 = Questionable
#
# N0ARY 1 #NOCAL.CA.USA.NA
#
+0RA 1 #SECO.CO.USA.NA
+0NCR 1 IA.USA.NA
+0WB8 1 #SEMI.MI.USA.NA
+0W7BZN 3 LE.0TW.EU
+DB0AAA 1 #BW.DEU.EU
+EA2AAA 1 EAZ.ESP.EU
```

There are three levels associated with the data in the files, SYSOP, USER and GUESS. A entry at the SYSOP level cannot be overridden by headers, incoming wp updates, or a user entry. It is considered golden. The USER level is header info for the bbss and either our user inputting the data or incoming updates with the U flag on. Guess is everything else.

Commands:

call

wp entry for user

NO, user XXXXX not in local white pages database

This command will both set the focus to the user supplied but also display the wp entry for the user in a form ready to be displayed to a user.

```
N0ARY
N0ARY @ N0ARY.#NOCAL.CA.USA.NA Bob 94086 Sunnyvale, CA
N6ZZZ
NO, user N6ZZZ not in local white pages database
```

CALL

call

NO, must set call first

Tells daemon to display call of user currently in focus.

[call] ADDRESS
[call] FNAME
[call] ZIP
[call] QTH
[call] HOME
[call] HLOC
[call] CHANGED

All of the proceeding commands are used to display the different fields associated to the user. The [call] is optional. If it is omitted the last user in focus is used.

N6ZFJ
N6ZFJ @ N0ARY.#NOCAL.CA.USA.NA Connie 94086 Sunnyvale, CA

ADDRESS
N0ARY.#NOCAL.CA.USA.NA

FNAME
Connie

ZIP
94086

QTH
Sunnyvale, CA

HOME
N0ARY

HLOC
#NOCAL.CA.USA.NA

CHANGED
749081327 752302439 On 931103 N6ZFJ/U @ N0ARY.#NOCAL.CA.USA.NA zip 94086 Connie Sunnyvale, CA

[call] SEEN [time_t]

This command marks the user as seen on the date/time supplied. If the date is not supplied than the current time is used.

[call] UPDATE

Mark the entry that an update has been sent out for this user.

CREATE call
KILL call
BBS CREATE call

Creation and deletion commands.

WRITE

Force database to be written to disk. This happens automatically every WPD_FLUSH_INTERVAL time which is normally set to 2-5 minutes.

UPLOAD <wp update entry>

This command is expecting a wp entry as it is received in the wp updates. All other formats will be ignored.

call level field value

level = SYSOP|USER|GUESS
field = FNAME|ZIP|HOME|HLOC|QTH

This is the syntax for editing the users fields. If the user in focus is not a bbs than editing the HLOC will generate an error.

Configuration Variables:

WPD_PORT	**44512**
WPD_USER_FILE	**wp/user.db**
WPD_BBS_FILE	**wp/bbs.db**
WPD_DUMP_FILE	**wp/dump.db**

These variables simply describe how to connect to wpd and tells wpd where it's databases are located.

WPD_GLOBAL_SERVER	**WP@N6QMY**
WPD_UPDATE_SUBJECT	**WP Update**
WPD_UPDATE_SIZE	**4096**
WPD_UPDATE	**1 day**
WPD_REFRESH	**14 days**
WPD_HOUR	**22**

Updates are sent the regional server every WPD_UPDATE period. An update will contain entries for all users that have had a field changed during that day. In addition to the direct changes we will send an entry for all users that have connected to the bbs in the last WPD_REFRESH period but have not had an update sent in in that period of time. The updates are sent to WPD_GLOBAL_SERVER and should have the subject WPD_UPDATE_SIZE. The sysop can configure how big the update messages should be. If a message exceeds WPD_UPDATE_SIZE it will be broken into multiple pieces. The daily updates go out at the hour specified by WPD_HOUR.

WPD_AGE	**6 months**
WPD_FLUSH	**1 minute**

The daemon will flush it's databases to disk every WPD_FLUSH if the image is dirty. The database it self is purged of old data as soon a it get WPD_AGE old.

BID Database (bidd)

/bbs/bin/b.bidd [-w|W] [-f] [-l|L] [-p bbsd_port] [-h bbsd_host] [-d#]

-w	**Write configuration dependencies**
-W	**Write configuration dependencies with opening comment**
-f filename	**Append configuration dependencies to filename, requires -w or -W**
-l	**Enable daemon logging, keep everything (debugging)**
-L	**Enable daemon logging, clear log file when daemon is idle (debugging)**
-p bbsd_port	**Where to find bbsd if different form the compiled in port**
-h bbsd_host	**Where to find bbsd if different from the compiled in host**
-d #	**Debugging flags**

The bid daemon is probably the simplest of all the daemons with the possible exception of logd. BIDs are used to keep from getting duplicate bulletins as they flood an area. Bids should be kept around much longer than the message itself. Some dups will take weeks to reach an area.

The bid database looks like this:

```
# v1 /bbs/message/bid.db
#
# This is a machine created file. If you edit it manually
# you need to kill the bid process, edit the file, then
# restart the bid daemon.
#
# This file is automatically updated once an hour if changes
# have been made to the runtime memory image.
#
# The '+' character at the beginning of the line indicates
# that the line has been preparsed so error detection can
# be skipped. If you edit or add a line delete the '+'.
#
# Entry format, non-parsed..
# BID SEEN
# |
# +-- mo/da/yr (02/28/93)
#
# 32405_N6QMY 02/28/93
#
+KL7Q046 753366068
+KL7Q047 753361870
+RACESBUL.300 753130977
+STS-5.016 753114764
+PGSW3308A 753089017
+GMSA3308A 753089009
+DBASE_001 753088978
+MWRN3308A 753088903
```

Commands:

bid

OK

NO, duplicate

This command checks to see if the BID is already in the database.

ADD bid

OK

Adds a new BID to the database

DELETE bid

OK

NO, bid not in database

Remove a bid from the database.

```
74000_N0ARY
OK

ADD 74000_N0ARY
OK

74000_N0ARY
NO, duplicate

DELETE 74000_N0ARY
OK

DELETE 74000_N0ARY
NO, bid not in database
```

Configuration Variables:

BIDD_PORT	**44514**
BIDD_FILE	**message/bid.db**
BIDD_FLUSH	**2 minutes**
BIDD_AGE	**3 months**

Internet Gateway Authentication Process (gated)

/bbs/bin/b.gated [-w|W] [-f] [-l|L] [-p bbsd_port] [-h bbsd_host] [-d#]

-w	**Write configuration dependencies**
-W	**Write configuration dependencies with opening comment**
-f filename	**Append configuration dependencies to filename, requires -w or -W**
-l	**Enable daemon logging, keep everything (debugging)**
-L	**Enable daemon logging, clear log file when daemon is idle (debugging)**
-p bbsd_port	**Where to find bbsd if different form the compiled in port**
-h bbsd_host	**Where to find bbsd if different from the compiled in host**
-d #	**Debugging flags**

The internet gateway allows those in the other networks to both send and receive packet mail as well as executing bbs commands via email messages. The key to the system is authenticating the user. Since internet has no restrictions on content we can gate freely from packet to internet. The other direction is not as open. For an internet user to be able to use the gateway he must send in a registration in advance. The registration contains info required for the WP entry as well as his internet address as found in the From: field. You must use the version found in the From: field no matter how ugly it is. This is the field that the gateway authentication code will compare to.

If an unregistered user attempts to use the gateway a nice bounce message is generated introducing him to the gateway and explaining how to register. One of the most common reasons for not being registered is that the registered users From: line has changed. A mail host may have been added, a different mailer used, a different uucp path, etc. To aid the user the gateway will do a search across the database looking for close addresses and those will be returned to the user as well, “Are any of these you?” type of thing.

The gate.db file looks like this:

```
# v1 /bbs/etc/gate.db
#
# This is a machine created file. If you edit it manually
# you need to kill the gated process, edit the file, then
# restart the gated daemon.
#
# This file is automatically updated once an hour if changes
# have been made to the runtime memory image.
#
N0ARY bob@hal.com 11/10/93 01/01/70
N0ARY N0ARY@bbs.Arasmith.COM 03/15/93 01/01/70
N0ARY bob@arasmith.com 06/02/93 01/01/70
N0ARY bob 11/14/93 01/01/70
N0ARY bob@suny 11/15/93 01/01/70
N6ZFJ connie 03/15/93 01/01/70
N6ZFJ arasmith@lams.msd.lmsc.lockheed.com 03/15/93 01/01/70
N6ZFJ connie@somewhere.usa 07/02/93 01/01/70
KA6ETB grafex!ka6etb 03/15/93 01/01/70
KA6ETB ka6etb@grafex.Cupertino.CA.US 09/17/93 01/01/70
KA6ETB steveh@grafex.Cupertino.CA.US 03/15/93 01/01/70
KA6ETB harding@nas.nasa.gov 03/15/93 01/01/70
```

Commands:

USER call

address_list

NO, user not found

Display all valid addresses for the supplied user. Any number of address maybe associated to a single call.

```
USER N0ARY
bob@hal.com
N0ARY@bbs.Arasmith.COM
bob@arasmith.com
bob
bob@suny
.

user n6zzz
NO, user not found
```

ADDRESS string

call

NO, address not found

Search for match for the supplied address. If found callsign of user is returned.

```
ADDRESS bob@hal.com
N0ARY

ADDRESS sam@bozo.com
NO, address not found
```

USER SEEN call

mo/da/yr

NO, user not found

Show the last time the supplied user was seen from any of his addresses.

ADDRESS SEEN string

mo/da/yr

NO, address not found

Show last time a specific address was seen.

```
USER SEEN N0ARY
11/15/93

USER SEEN N6ZZZ
NO, user not found

ADDRESS SEEN BOB@HAL.COM
11/10/93

ADDRESS SEEN TOM
NO, address not found
```

ADD call address

OK

NO, address already in database

Add a callsign address pair to the database

```
ADD N0ARY bob@voilet.arasmith.com
OK

ADD N0ARY BOB
NO, address already in database
```

DELETE call address

OK

NO, call/address pair not found

Remove a call address pair from the database

```
DELETE N0ARY bob@voilet.arasmith.com
OK

DELETE N0ARY bob@arrow.arasmith.com
NO, call/address pair not found
```

TOUCH address

OK

NO, address not found

Refresh the address supplied.

GUESS address

When an address doesn't match any of the entries in the database this command will give you possible close addresses. It works by analyzing what you give it as an address. If the address starts with a letter than it tries to match to usernames. If the address begins with an '@' then the daemon strips off all but the xxxx.xxx portion and compares with that.

```
GUESS bob@arasmith.com
Possible matches on username:
N0ARY bob@hal.com
N0ARY bob@arasmith.com
N0ARY bob
N0ARY bob@suny
VK3VO bob@mullara.met.unimelb.EDU.AU
WA7OVU bobh@hpesoc1.cup.hp.com

Possible matches on domain:
N0ARY N0ARY@bbs.Arasmith.COM
N0ARY bob@arasmith.com
N6RME paulb@km6px.Arasmith.COM
.
```

WRITE

OK

This command simply forces a write of the gateway database to disk whether it is clean or dirty. The daemon will save to disk on it's own every few minutes if the image is dirty.

Configuration Variables:

GATED_PORT	**44515**
GATED_FILE	**etc/gate.db**
GATED_AGE_WARN	**4 months**
GATED_AGE_KILL	**5 months**

Every week the database is scanned for addresses that haven't been seen in GATED_AGE_WARN. When an old address is encountered a message is sent to the owner informing him that the registration is about to expire. Sending a message through the gateway is all that is required to refresh the entry. If the record is still unused after GATED_AGE_KILL the entry will be deleted. If the user has no more entries associated with his call and he has not checked into the bbs via another method his user account will be deleted as well.

BBS Logging Process (logd)

/bbs/bin/b.gated [-w|W] [-f] [-l|L] [-p bbsd_port] [-h bbsd_host] [-d#]

-w	**Write configuration dependencies**
-W	**Write configuration dependencies with opening comment**
-f filename	**Append configuration dependencies to filename, requires -w or -W**
-l	**Enable daemon logging, keep everything (debugging)**
-L	**Enable daemon logging, clear log file when daemon is idle (debugging)**
-p bbsd_port	**Where to find bbsd if different form the compiled in port**
-h bbsd_host	**Where to find bbsd if different from the compiled in host**
-d #	**Debugging flags**

The logd process is used to buffer up log information while a bbs forwarding process is running and append it to the end of the log file when the process closes. This keeps the single forwarding log file from receiving interspersed updates from multiple processes. It also guarantees the log entry will be made, even if the forwarding process should have a critical failure.

This daemon has no commands. The first string it expects to see is a who-am-i string that it will use to stamp the log file. From then on every string it receives is simply added to the buffer. When the client process disappears the entire buffer is written to the log file.

Configuration Variables:

LOGD_PORT	**44511**
LOGD_FILE	**logd/bbslog**
LOGD_DIR	**logd**